Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 137 (2016) 180 – 185

Procedia Engineering

www.elsevier.com/locate/procedia

GITSS2015

An Approximation Algorithm for the Minimum Vertex Cover Problem

Jingrong Chen*, Lei Kou, Xiaochuan Cui

School of Mathematics and Physics, Lanzhou Jiaotong University, Lanzhou 730070, P. R. China

Abstract

The minimum vertex cover problem is a basic combinatorial optimization problem. Given an undirected graph the objective is to determine a subset of the vertices which covers all edges such that the number of the vertices in the subset is minimized. In the paper, based on Dijkstra algorithm, an approximation algorithm is obtained for the minimum vertex cover problem. In the process of getting a vertex cover, the maximum value of shortest paths is considered as a standard, and some criteria are defined. The time complex of the Algorithm is $O(n^3)$, where n is the number of vertices in a graph. In the end, an example is given to illustrate the process and the validity of the Algorithm.

Peer-review under responsibility of the Department of Transportation Engineering, Beijing Institute of Technology

Keywords: minimum vertex cover problem; Dijkstra algorithm; approximation algorithm; time complexity

1. Introduction

The minimum vertex cover problem is one of fundamental NP-hard problems in the combinatorial optimization [11], and it has received a lot of attention these last decades. To find a minimum vertex cover is very difficult, but to find an alternative is easier. So, many approximation algorithms have been proposed to construct vertex cover in different ways. Such an algorithm must have a polynomial time complexity and must return a solution. And the running time and the quality of the solution are considered to evaluate an approximation algorithm. The quality is traditionally measured in terms of approximation ratio of the solution and the optimal one. The best constant approximation ratio known is 2, and this is the best-known worst-case one [8]. Approximation ratio is not necessarily a constant. If approximation ratio is small, the quality is good. There are several works have been

* Corresponding author. Tel.: 86-13639360619.
E-mail address: chenjr@mail.lzjtu.cn

Peer-review under responsibility of the Department of Transportation Engineering, Beijing Institute of Technology
doi:10.1016/j.proeng.2016.01.248

devoted to this subject [10, 14, 20] .

In order to get an approximation algorithm, many methods have been proposed, including direct algorithms, parameterized algorithms, and intelligent algorithms, etc..

For direct algorithms, there are the Maximum Degree Greedy algorithm [7], the Depth First Search (DFS) algorithm [16], the Edge Deletion (ED) algorithm [12], the ListLeft (LL) algorithm [1], the ListRight (LR) algorithm [9], the Iterated Local Search algorithm [21] and etc.. Some improvements for classical algorithms and calculation techniques are used in the algorithms mentioned above.

For a graph G and a given parameter k, the parameterized vertex cover problem is to get a vertex cover of G with at most k vertices [3]. In 2001, Chen and Kanj introduced some new properties and techniques for the vertex cover problem, and obtained an improved algorithm with time complexity $O(1.2852^k + kn)$ [4]. Exponential algorithms have also been proposed [5,6,15], among which the best running time is $O(1.2745^k k^4 + kn)$ [5]. In 2010, Chen etc. presented an algorithm with running time bounded by $O(1.2738^k + kn)$, which is a significant improvement over the previous polynomial algorithms [3].

For intelligent algorithm, Singh etc. proposed a hybrid approach for the problem combining with Steady-state Genetic algorithm and Greedy Heuristic [17]. Xu etc. presented an efficient Simulated Annealing algorithm and simulated on several benchmark graphs [22]. Stefan etc. applied a modified Reactive Tabu Search Approach with Simulated Annealing for the minimum weight vertex cover problem [19]. Harsh applied the theory of Natural Selection via Genetic algorithms for solving the problem [2]. Ant Colony Optimization algorithm is also used to discuss the problem [13, 18].

In this paper, we propose an approximation algorithm for the minimum vertex cover problem based on Dijkstra algorithm. There is no restriction on graph and degree of vertex. Some simple techniques are introduced for computing. The maximum value of shortest paths is considered as a standard for getting a vertex cover, and some criteria are defined. The time complex of the Algorithm is $O(n^3)$, where n is the vertex number of a graph. In the end, an example is given to illustrate the process and the validity of the Algorithm.

2. The Algorithm

Let $G = (V, E)$ be a graph, where V is the set of vertices and E is the set of edges. A vertex cover S is a subset of V such that each edge has at least one ends in S. A minimum vertex cover of G is one subset of V in which the number of vertices is minimized. In this paper, we concern with undirected and connected plane graphs. If G is disconnected, each components would be discussed separately.

Label all vertices of G, and let an arbitrary vertex be the initial point denoted by u_1. Now we have set $S = \{u_1\}$, subgraph $H = G - S$, allowed set $L = V - S - \{\text{the isolated vertices of } H\}$.

Assume we already have the set $S = \{u_1, u_2, \ldots, u_{i-1}, u_i\}$. If subgraph H are consist of isolated vertices, then S is a vertex cover of G. Because in graph G there is no edge with two ends in H. Now the edge set E is partitioned into two sets E_1 and E_2 with $E = E_1 \cup E_2$, where the edges of E_1 have one endpoint in S, the edges of E_2 have both endpoints in S. Now each edge of G has at least one ends in set S, so S is a vertex cover of G.

If there are complete graphs or circles or paths in the connected components of H, incorporate their corresponding minimum covers into the set S separately, and put all adjacent vertices of leaves into the set S also.

If u_i is an isolated vertex in $H = G - \{u_1, u_2, \cdots, u_{i-1}\}$, then let u_{i+1} be an arbitrary vertex of L and $S = S \cup \{u_{i+1}\}$. Otherwise, by using Dijkstra algorithm in subgraph $H = G - \{u_1, u_2, \cdots, u_{i-1}\}$, we can get the shortest distances $d_{u_i v} (v \in L)$ from vertex u_i to all vertices of set L. Let the maximum value

$d = \max_{v \in L} \{d_{u_i v}\}$.

Then according to the two cases below to get a vertex cover of G.

Case 1: $d > 1$

Let $T=\{v \mid d_{u_i v}=d-1, v \in L\}$, which is a set containing the vertices with the distance of $d-1$ from u_i to them. According to the following criteria, we will get the vertex u_{i+1}.

Delete each vertex of set T separately in subgraph H. Let u_{i+1} be the vertex which has the minimum connected component increase after deleting, and the count of connected components does not include the isolated vertices.

If they have the same increase of connected components, then let u_{i+1} be the vertex with maximum degree in H.

If there is only one vertex in T, then let it be u_{i+1}.

If the conditions of all vertices in T are the same after deleting, then let u_{i+1} be the one with the minimum label.

Let $S=S \cup \{u_{i+1}\}$, and repeat the process above.

Case 2: $d=1$

It is obviously that u_i is adjacent to each vertex of set L. If $H=G-S=G-\{u_1,u_2,\cdots,u_i\}$ is the disjoint union of complete graphs or circles or paths, then it is easy to get a minimum vertex cover of H. So we obtain a vertex cover of G. Otherwise, let u_{i+1} be the vertex with the maximum degree, or the minimum label if they have same degree in $H=G-S=G-\{u_1,u_2,\cdots,u_{i-1},u_i\}$. Now let $S=S \cup \{u_{i+1}\}$, and repeat the process above.

Now list the specific steps of the Algorithm below.

The Algorithm

Step 1. Initialization: initialize the node labels and $S=\{u_1\}$, $L=\varnothing$, $T=\varnothing$.

Step 2. $S=\{u_1,u_2,\cdots,u_i\}$

If subgraph $H=G-S$ are consist of isolated vertices, then **stop the Algorithm**.

If there are complete graphs or circles or paths in the connected components of H, then incorporate their corresponding minimum cover into the set S separately, and put all adjacent vertices of leaves into the set S also. Go to Step 2 and still use the vertex u_i to search.

Otherwise, go to Step 3.

Step 3. Let $H=G-\{u_1,u_2,\cdots,u_{i-1}\}$ and $L=V-S-\{\text{the isolated vertices of } H\}$.

If u_i is an isolated vertex in H, then let u_{i+1} be an arbitrary vertex of L. $S=S \cup \{u_{i+1}\}$, go to Step 2.

Else invoke Dijkstra algorithm to get the shortest distances $d_{u_i v}(v \in L)$ in H and compute maximum value d. Go to Step 4.

Step 4. If $d>1$, go to Step 5, else Step 6.

Step 5. let $T=\{v \mid d_{u_i v}=d-1, v \in L\}$, then according to Case 1 in the text to select u_{i+1}. let $S=S \cup \{u_{i+1}\}$ and go to Step 2.

Step 6. If subgraph $H=G-S=G-\{u_1,u_2,\cdots,u_i\}$ are the disjoint union of complete graphs or circles or paths, then a minimum vertex cover of H is obtained and **stop the Algorithm**.

Otherwise, let u_{i+1} be the vertex with the maximum degree, or with the minimum label if they have same degree in H. Let $S=S \cup \{u_{i+1}\}$ and go to Step 2. □

Proposition 1. *Set S is a vertex cover of graph G at the end of the Algorithm.*

Proof. Assume that S is not a vertex cover of G at the end of the Algorithm. Let $H=G-S$, then there is at least one edge whose two ends are not in the S. According to stop conditions of the Algorithm at Step 2 and Step 6, H are consist of isolated vertices, or the disjoint union of complete graphs or circles or paths, but H is neither. So according to the performance of the Algorithm, the Algorithm should not be ended. This is a contradictory, then the result is proved. □

Proposition 2. *The time complexity of the Algorithm is at most $O(n^3)$.*

Proof: We just to count the number of operations in each step. In Step 1, $n+3$ assignments are needed. In Step 2,

there are at least n^2 comparisons to judge whether H are consist of isolated vertices by using the adjacent matrix of H. There are at most n^2 comparisons to judge whether H are the disjoint union of complete graphs or circles or paths. Add up the entries of each line in adjacent matrix to judge whether there are one-degree vertices, then at most n^2-n additions and n comparisons are generated. In step 3, the time complexity of Dijkstra algorithm is $O(n^2)$, and to get the maximum d generate $n-1$ comparisons. In Step 5, there are $n-1$ comparisons to get set T, and $2(n-1)$ comparisons to obtain vertex u_{i+1}. In Step 6, there are n^2+n-1 comparisons. Therefore, the total time complexity of the Algorithm is about $O[(n-1)\times(4n^2+5n-5)]\approx O(n^3)\sqrt{b^2-4ac}$ by approximating and simplifying the calculation. □

3. Example

Now we use the example given in Fig. 1 to illustrate the concrete process of the Algorithm, where the weight of each edge is assigned 1.

Step 1. Initializing the node labels, and let $S=\{v_6\}$, $L=T=\varnothing$.

Step 2. Vertex v_2 is a leafage in $H=G-S=G-\{v_6\}$, then $S=\{v_6,v_8\}$ and go to Step 3.

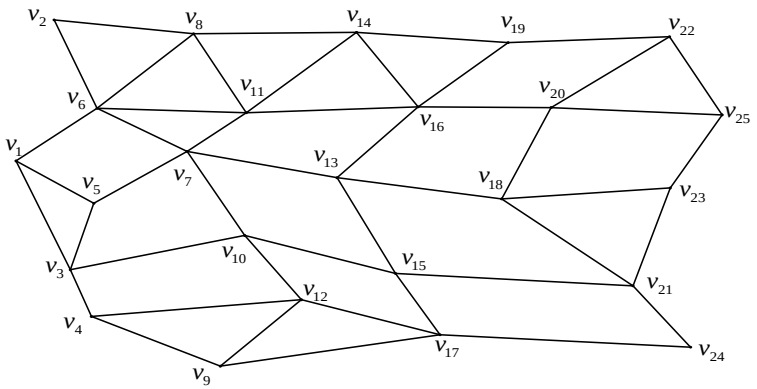

Fig. 1. Graph G

Step 3. Allowed set

$$L=V-S-\{\text{the isolated vertices of } H\}=V-\{v_6,v_8\}-\{v_2\}=\{v_i, i\neq 2,6,8\}.$$

Invoking Dijkstra algorithm to get the shortest distance $d_{6,j}$ from v_6 to each vertex of L, which are listed in Table 1, and we get $d=5$. Go to Step 4.

Step 4. $d>1$, go to Step 5.

Table 1. The shortest distances from v_6 to each vertex of L

$d_{6,j}$	v_1	v_3	v_4	v_5	v_7	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
v_{17}	1	2	3	2	1	4	2	1	3	2	2

$d_{6,j}$	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}	v_{21}	v_{22}	v_{23}	v_{24}	v_{25}
v_{17}	3	2	4	3	3	3	4	4	4	5	4

Step 5. $T=\{v_9,v_{17},v_{21},v_{22},v_{23},v_{25}\}$. The number of connected components are same after deleting each vertex of T. Vertex v_{17} meets the requirements, then $S=\{v_6,v_8,v_{17}\}$. Go to Step 2

Step 2. Vertex v_{24} is a leafage in $H = G - S$, then $S = \{v_6, v_8, v_{17}, v_{21}\}$ and go to Step 3.

Step 3. Allowed set

$L = V - S - \{\text{the isolated vertices of } H\} = \{v_i, i \neq 2, 6, 8, 17, 21, 24\}$.

Invoking Dijkstra algorithm to get the shortest distance $d_{17,j}$ from v_{17} to each vertex of L, which are listed in Table 2, and we get $d = 5$.

Repeat the above steps until the Algorithm is stopped.

Table 2. The shortest distances from v_{17} to each vertex of L

$d_{17,j}$	v_1	v_3	v_4	v_5	v_7	v_7	v_{10}	v_{11}	v_{12}	v_{13}
v_{17}	4	3	2	4	3	1	2	4	1	2

$d_{17,j}$	v_{14}	v_{15}	v_{16}	v_{18}	v_{19}	v_{20}	v_{22}	v_{23}	v_{25}
v_{17}	4	1	3	3	4	4	5	4	5

In Table 3, we list several vertex covers of G with different vertices as initial points which are selected arbitrarily. Set S_{24} has the minimum vertices, so let it be a vertex cover of G. And we also find that a couple of vertices, such as v_1, v_2, v_4, v_{10} etc., exist only in one vertex cover. So let them be the initial point separately to get vertex cover of G, which are listed in Table 4. The number of vertex cover sets are all 15, so different initial points can get good results.

Table 3. Computational results

Initial points	Vertex cover set	Number of vertex
v_2	$S_2 = \{v_1, v_2, v_3, v_4, v_7, v_8, v_{11}, v_{12}, v_{15}, v_{16}, v_{17}, v_{18}, v_{19}, v_{20}, v_{21}, v_{25}\}$	16
v_3	$S_3 = \{v_3, v_5, v_6, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}, v_{17}, v_{18}, v_{19}, v_{20}, v_{21}, v_{25}\}$	16
v_6	$S_6 = \{v_3, v_5, v_6, v_7, v_8, v_9, v_{11}, v_{12}, v_{15}, v_{16}, v_{17}, v_{18}, v_{19}, v_{20}, v_{21}, v_{25}\}$	16
v_9	$S_9 = \{v_3, v_5, v_6, v_7, v_8, v_9, v_{12}, v_{14}, v_{15}, v_{16}, v_{17}, v_{18}, v_{19}, v_{20}, v_{21}, v_{25}\}$	16
v_{14}	$S_{14} = \{v_3, v_5, v_6, v_7, v_8, v_9, v_{12}, v_{14}, v_{15}, v_{16}, v_{17}, v_{18}, v_{20}, v_{21}, v_{22}, v_{25}\}$	16
v_{24}	$S_{24} = \{v_3, v_5, v_6, v_7, v_8, v_9, v_{12}, v_{14}, v_{15}, v_{16}, v_{18}, v_{20}, v_{22}, v_{23}, v_{24}\}$	15

Table 4. Computational results

Initial points	Vertex cover set	Number of vertex
v_1	$S_1 = \{v_1, v_3, v_6, v_7, v_8, v_9, v_{12}, v_{14}, v_{15}, v_{16}, v_{17}, v_{18}, v_{21}, v_{22}, v_{25}\}$	15
v_4	$S_4 = \{v_3, v_4, v_5, v_6, v_7, v_8, v_{12}, v_{13}, v_{14}, v_{16}, v_{17}, v_{20}, v_{21}, v_{22}, v_{23}\}$	15
v_{10}	$S_{10} = \{v_3, v_4, v_5, v_6, v_8, v_9, v_{10}, v_{11}, v_{13}, v_{14}, v_{17}, v_{20}, v_{21}, v_{22}, v_{23}\}$	15

4. Conclusion

In this paper, we have proposed an approximation algorithm for the minimum vertex cover problem based on Dijkstra algorithm. The Algorithm is easy to implement, and there is no special requirement on the degree of vertices. We get different vertex covers with different initial points, so it would be good to pick up a couple of points to get different results. If we make each vertex of G as an initial point, then n vertex covers would be obtained. Select the set with the least vertices as an approximation minimum vertex cover of G. So the time complex of the Algorithm would be at most $O(n^4)$.

In future, we would investigate the minimum weight vertex cover problem. Because the longest shortest-path is used as a standard in the Algorithm, the results would be different for the graph with weighted edges. Furthermore, we will study the problem in which both edges and vertices are weighted.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No.61463027), Ph.D. Programs Foundation of Ministry of Education of China (No.20136204120007), Fundamental Research Funds of Gansu Province (No.212092-1).

References

[1]Avis D., Imamura T., 2007. A list heuristic for vertex cover, Operations Res. Lett. 35, 201-204.
[2]Bhasin H., 2012. Harnessing genetic algorithm for vertex cover problem. International Journal on Computer Science and Engineering, 4,218-223.
[3]Chen J., Kani I.A., Xia G., 2010. Improved upper bounds for vertex cover, Theoretical Computer Science, 411,3736-3756.
[4]Chen J., Kanj I.A., Jia W., 2001. Vertex cover: further observations and further improvements, Journal of Algorithms 41,280-301.
[5]Chandran L., Grandoni F., 2005. Refined memorisation for vertex cover, Inform. Proc. Lett., 93, 125-131.
[6]Chen J., Liu L., Jia W., 2000. Improvement on vertex cover for low degree graphs, Networks, 35,253-259.
[7]Cormen T.H., Leiserson C.E., Rivest R.L., Andstein C., 2009. Introduction to Algorithms 3rd Ed. MIT Press, Cambridge, MA.
[8]Delbot F., Laforest C., 2010. Analytical and experimental comparison of six algorithms for the vertex cover problem. ACM Journal of Experimental Algorithmics, 15(1.4),1-26.
[9]Delbot F., Andlaforest C., 2008. A better list heuristic for vertex cover, Inform. Proc. Lett.,107, 125-127.
[10]Dinur I., Ansafra S., 2005. On the hardness of approximating minimum vertex-cover, Ann. Math., 162,439-485.
[11]Garey M.R., Johnson D.S., 1978. Strong NP-completeness results: motivation, examples, and implications, Journal of ACM, 25, 499-508.
[12]Garey, M.R., Andjohnson D., 1979. Computers and Intractability, Freeman and Co., New York.
[13]Jovanovic R., Tuba M., 2011. An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem. Applied Soft Computing, 11,5360-5366.
[14]Karakostas G., 2009. A better approximation ratio for the vertex cover problem. ACM Transactions on Algorithms, 5, 41:1-8.
[15]Niedermeier R., Rossmanith P., 2003. On efficient fixed-parameter algorithms for weighted vertex cover, Journal of Algorithms, 47,63-77.
[16]Savage C., 1982. Depth-first search and the vertex cover problem, Inform. Proc. Lett., 14,233-235.
[17]Singh A., Gupta A.K., 2006. A hybrid heuristic for the minimum weight vertex cover problem, Asia-Pacific Journal of Operational Research, 23,273-285.
[18]Shyu S.J., Yin P.Y., MT Lin B., 2004. An ant colony optimization algorithm for the minimum weight vertex cover problem. Annals of Operations Research, 131,283-304.
[19]Voß S., Fink A., 2012. A hybridized tabu search approach for the minimum weight vertex cover problem. Journal of Heuristics 18, 869-876.
[20]Vazirani V., 2002. Approximation algorithms, Springer, Berlin.
[21]Witt C., 2012. Analysis of an iterated local search algorithm for vertex cover in sparse random graphs, Theoretical Computer Science 425,117-125.
[22]Xu X., Ma J., 2006. An efficient simulated annealing algorithm for the minimum vertex cover problem, Neurocomputing, 69,913-916.